EECS C145B / BioE C165: Image Processing and Reconstruction Tomography

Lecture 9 (Revision II)

Jonathan S. Maltz

This handout contains copyrighted material. It is for personal educational use only. Do not distribute.

jon@eecs.berkeley.edu
http://muti.lbl.gov/145b
510-486-6744

Topics to be covered

1. Basic concepts in tomography
2. Non-diffracting radiation
3. x-ray contrast and basic physics
4. x-ray projections
5. The Radon transform
6. The sinogram
7. Properties of the Radon transform
8. Review of polar coordinates
9. The Fourier slice theorem
10. The backprojection operator
11. Backprojection of filtered projections

Reading

Assigned reading:

- Course Reader, pp. 6-11, 77-87.

Additional reading:

- Cho, “Foundations of Medical Imaging”, John Wiley and Sons (1993), Chapter 3.
- Bracewell pp. 517-538.

Advanced reading:

- Natterer, “The Mathematics of Computerized Tomography”, John Wiley and Sons (1986), Chapters I and II.

Tomography with non-diffracting sources

Definition: **Tomography** is the process of taking measurements on the outside of a body and using these measurements to construct cross-sectional images of the inside of that body.

Etymology: The Greek word “tomos” means “section”. Tomography allows us to visualize the cross-sections of a body without cutting open and sectioning the body.

Definition: **Computed tomography (CT)** involves the taking of measurements outside a body that are then fed to a computer that reconstructs cross-sectional images of the inside of that body (This term was coined in 1975).

Definition: **Non-diffracting** radiation travels through a specific medium along straight lines.

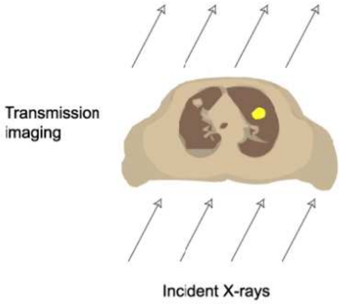
Examples of radiation that travel in straight lines (to good approximation) are **x-rays and gamma rays** through the entire human body, and **ultrasound** through soft tissue such as the breast.

Transmission and emission tomography

In **transmission tomography** radiation from **sources outside the body** is directed inside the body. Detectors on the outside of that body measure the radiation that comes out of that body as a consequence of the applied radiation. Using knowledge of the applied and measured radiation, cross-sectional images are constructed.
Examples:

1. x-ray computed tomography (x-ray CT)
2. Magnetic resonance imaging (MRI)
3. Ultrasound tomography
4. Optical fluorescence imaging

Transmission and emission tomography



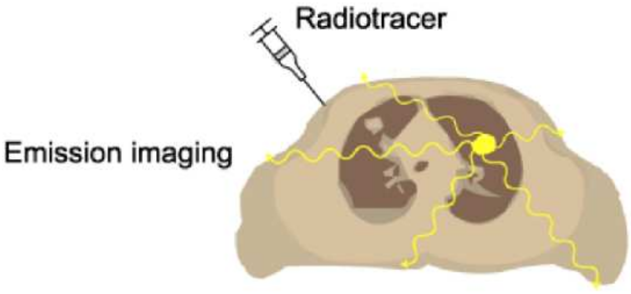
In **transmission tomography**, incident x-rays are attenuated by matter within the body. As a result, the amount of radiation that emerges from the body varies in intensity depending on the amount of the material the radiation traveled through, and the density of the material traversed. Contrast is based on the x-ray attenuation coefficients of the matter in the cross-section (e.g. bone attenuates x-rays much more than fat, so the two have strong contrast in x-ray CT images).

Transmission and emission tomography

In **emission tomography**, radiation emanating from **sources inside a body** is detected by sensors outside that body. Cross-sectional images of the distribution of these sources are created based on these measurements.
Examples:

1. Single photon emission computed tomography (SPECT) and positron emission tomography (PET): Radioactive tracers that emit gamma rays are injected into the body. Radiation detectors intercept and count gamma photons that exit the body.
2. Optical bioluminescence imaging: A chemical that produces light when it comes into contact with a second chemical is injected. When the organism produces the second chemical, an exothermic reaction occurs. Light produced in this way inside the body of the organism is recorded by sensitive optical cameras. The organism must be in a light-tight box.

Transmission and emission tomography

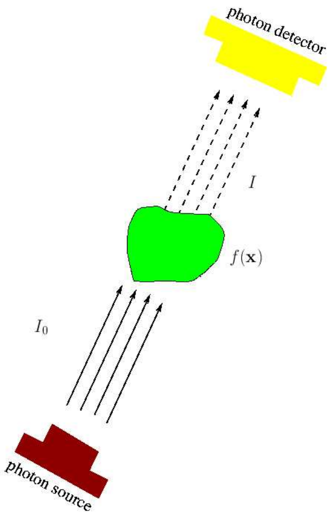


In **emission tomography**, an injected radiotracer emits gamma photons. These photons can be detected after they emerge from the body. Using tomographic reconstruction algorithms, the distribution of radiotracer in the body can be mapped. Contrast is based on spatial differences in radiotracer concentration.

Tomography of non-diffracting and diffracting sources

- In this course, we will study mathematical methods for the tomographic reconstruction of data produced by **non-diffracting** sources.
- Tomographic reconstruction of complicated distributions (such as animals) that are imaged using **diffracting** radiation is typically a **non-linear**, **underdetermined** problem (more unknowns that independent measurements). Until recently, the solution of this type of reconstruction problem has been computationally infeasible. Methods for the solution of these problems are beyond the scope of this course.
- We will see that basic reconstruction of cross-sections from projections is **linear** problem. Typically, we can always take more independent measurements that we have unknowns. As a result, this problem is usually **overdetermined**.

Introduction to x-ray CT physics



In x-ray CT, we have a photon source that produces (in this case) a parallel incident beam of uniform intensity I_0. We assume that the x-rays travel along straight lines. The intensity I of an emerging ray depends on the length of its line of travel through the object $f(\mathbf{x})$, and the material properties of $f(\mathbf{x})$ along this line of travel.

x-ray projection physics

- Let $\mu(\mathbf{x})$ represent the attuenutation coefficient at a point $\mathbf{x}$. Then the intensity of an emerging beam I can be expressed in terms of an incident beam of intensity I_0 as follows:

  $$I = I_0 \, \mathrm{e}^{-\int_L \mu(\mathbf{x})\, du}$$

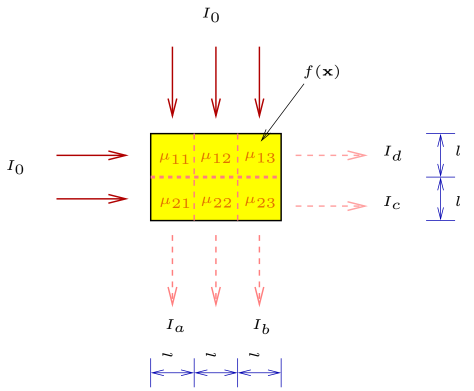
  where L is the line of travel and u is the distance along this line.
- The units of $\mu(\mathbf{x})$ are $(\text{distance})^{-1}$ or "fractional attenuation per meter" (m^{-1}) in the metric system.
- Another unit often used is the *Hounsfield unit* (HU). A material that has an attuenuation coefficient of 1 HU attenuates beam intensity 1000 times more per unit length than water.

x-ray projections: Discrete case

In the general discrete case, when a ray travels through K materials, each having attenuation coefficient μ_k, and has an interaction length of l_k with the kth material, the emerging intensity is given by:

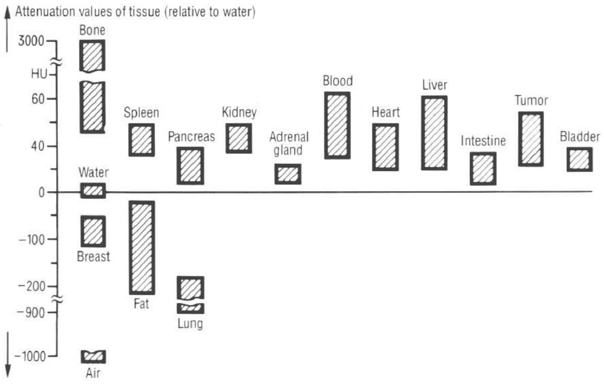
$$I = I_0 \, \mathrm{e}^{-\sum_{k=1}^{K} \mu_k \, l_k}$$

x-ray projections: Discrete example



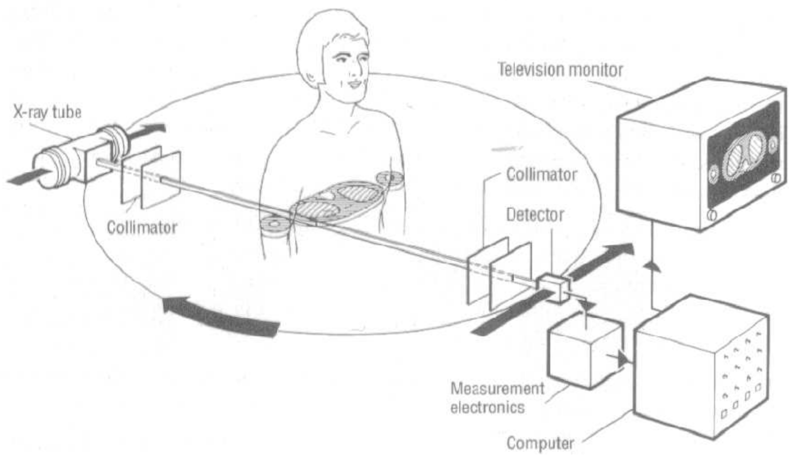
$$
\begin{aligned}
I_a &= \left[\left[I_0\,\mathrm{e}^{-\mu_{11}l}\right]\mathrm{e}^{-\mu_{21}l}\right] = I_0\,\mathrm{e}^{-(\mu_{11}l+\mu_{21}l)} \\
I_d &= \left[\left[\left[I_0\,\mathrm{e}^{-\mu_{11}l}\right]\mathrm{e}^{-\mu_{12}l}\right]\mathrm{e}^{-\mu_{13}l}\right] = I_0\,\mathrm{e}^{-(\mu_{11}l+\mu_{12}l+\mu_{13}l)} \\
I_c &= I_0\,\mathrm{e}^{-\sum_{k=1}^{3}\mu_{1k}\,l} \\
I_b &= I_0\,\mathrm{e}^{-\sum_{k=1}^{2}\mu_{k3}\,l}
\end{aligned}
$$

x-ray projection physics



Many body structures and tissues possess attenuation coefficients that do not overlap. Attenuation coefficient differences produce differential contrast in attenuation maps of the distribution. It follows that these structures and tissues may easily resolved as separate entities in an x-ray CT image. Figure due to Linke (1990).

x-ray projection physics



Typical configuration of X-ray CT imaging instrumentation. Figure due to Linke (1990).

x-ray CT: History

1917 Johan Radon (1887-1956), a Bohemian mathematician, publishes a paper that establishes the mathematical foundations for tomography.

1963 Allan Cormack at Tufts University popularizes the idea of x-ray CT, but does not build a practical scanner.

1972 First practical scanner built by Hounsfield at EMI in England.

1979 Cormack and Hounsfield receive Nobel Prize for their contributions to this field.

x-ray projections

- The emerging intensity I is a non-linear function of $\mu(\mathbf{x})$:

$$\begin{aligned} I &= I_0\,\mathrm{e}^{-\int_L \mu(\mathbf{x})\,du} && \text{continuous case, or} \\ I &= I_0\,\mathrm{e}^{-\sum_{k=1}^{K} \mu_k\, l_k} && \text{discrete case} \end{aligned}$$

- We can transform I into a linear function of $\mu(\mathbf{x})$ by defining a single **x-ray projection measurement** as

$$p = \ln \frac{I_0}{I}$$

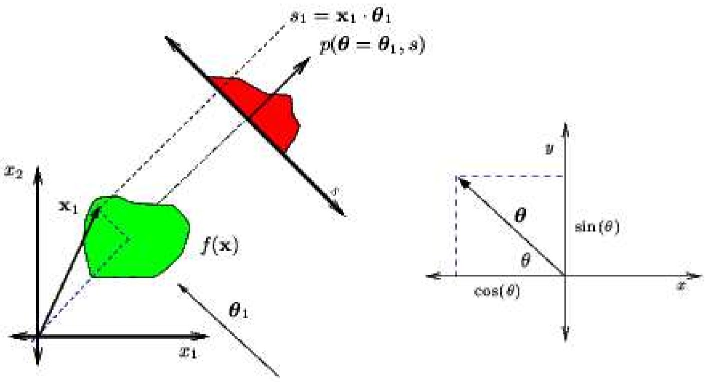
- For multiple rays, the function describing the collection of all projections is

$$\begin{aligned} p \triangleq p(\boldsymbol{\theta}, s) &= \int_L \mu(\mathbf{x})\,du \\ &= \int_{\mathbf{x}\cdot\boldsymbol{\theta}=s} \mu(\mathbf{x})\,d\mathbf{x} \end{aligned}$$

  where $\boldsymbol{\theta}$ is a unit vector perpendicular to the ray direction. The quantity s is the radial distance from the origin along $\boldsymbol{\theta}$.

- The function $p(\boldsymbol{\theta}, s)$ is called the **Radon transform** of $\mu(\mathbf{x})$.

The Radon transform



Geometry of the Radon transform $\mathbf{R}\,f(\mathbf{x}) = p(\boldsymbol{\theta}, s)$ of a 2D distribution $f(\mathbf{x})$. In x-ray CT $f(\mathbf{x}) \triangleq \mu(\mathbf{x})$. The projection at $\boldsymbol{\theta} = \boldsymbol{\theta}_1$ is shown. The right diagram shows the components of the unit vector $\boldsymbol{\theta}$ in terms of the 2D Cartesian coordinates $x \triangleq x_1$ and $y \triangleq x_2$.

The Radon transform

We will formalize the Radon transform in both vector form (valid for N-dimensional distributions $f(\mathbf{x})$), and in scalar form for 2D distributions $f(x, y)$.

- The N-dimensional Radon transform may be described as:

$$p(\boldsymbol{\theta}, s) = \mathbf{R}f(\mathbf{x}) = \int_{\mathbf{x}\cdot\boldsymbol{\theta}=s} f(\mathbf{x})\,d\mathbf{x} = \int_{\Re^N} f(\mathbf{x})\,\delta(\mathbf{x}\cdot\boldsymbol{\theta} - s)\,d\mathbf{x}$$

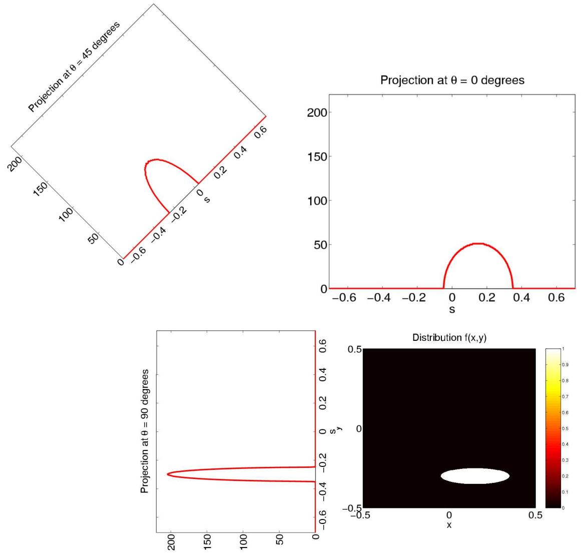
  where $\mathbf{x} \in \Re^N$. The geometry is described in the previous figure.

- For $\mathbf{x} \in \Re^2$ we have:

$$\begin{aligned} p(\boldsymbol{\theta}, s) &= \mathbf{R}f(x, y) = \int_{\left[\begin{smallmatrix} x \\ y \end{smallmatrix}\right]\cdot\boldsymbol{\theta}=s} f(x, y)\,dx\,dy \\ &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x, y)\,\delta\Big(x\,\cos(\theta) + y\,\sin(\theta) - s\Big)\,dx\,dy \end{aligned}$$

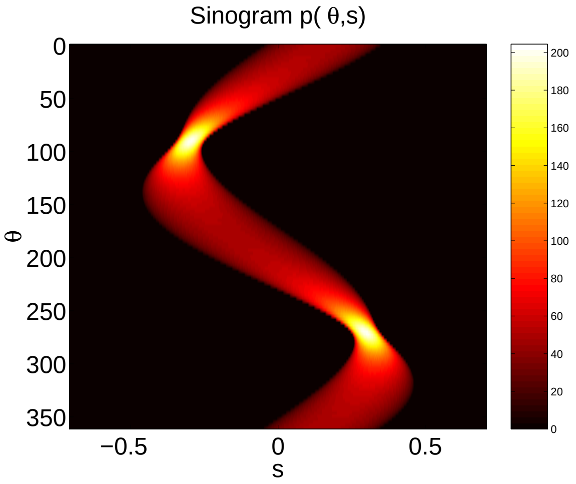
The Radon transform

The Radon transform of a 2D distribution $f(x, y)$ is a set of line integrals called projections $p(\boldsymbol{\theta}, s)$. Shown here are three projections of $f(x, y)$, each at a different angle θ with respect to the x-axis. Note th



The Sinogram

The set of projection measurements $p(\boldsymbol{\theta}, s)$ is often visualized as an image called the **sinogram**. Each row of the sinogram is an image representation of a projection.



The Sinogram: Properties

- The sinogram is periodic with a period of 360 degrees.
- Only half of the period of the sinogram is unique. Why?

- The sinogram get its name from the fact that each point in the x-y plane becomes a sinusoid in the θ-s plane. To see this we take the Radon transform of $\delta(x - x_0, y - y_0)$:

$$\begin{aligned} p(\boldsymbol{\theta}, s) &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)\,\delta\Big(x\,\cos(\theta) + y\,\sin(\theta) - s\Big)\,dx\,dy \\ &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \delta(x - x_0,\, y - y_0)\,\delta\Big(x\,\cos(\theta) + y\,\sin(\theta) - s\Big)\,dx\,dy \\ &= \delta\Big(x_0\,\cos(\theta) + y_0\,\sin(\theta) - s\Big) \end{aligned}$$

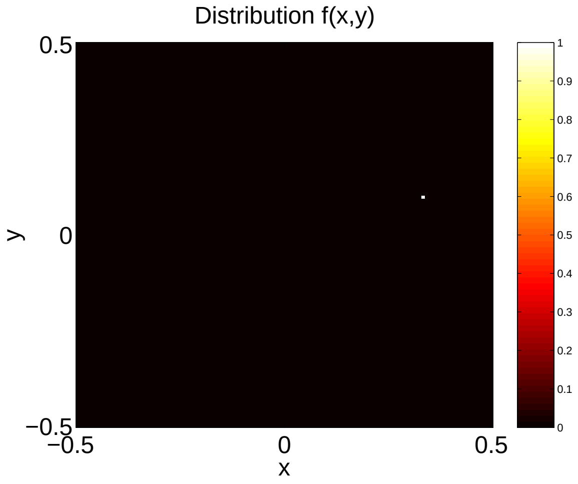
Thus the point (x_0, y_0) gets mapped to the curve:

$$s = x_0\,\cos(\theta) + y_0\,\sin(\theta)$$

which is a sinusoidal function of $\boldsymbol{\theta}$.

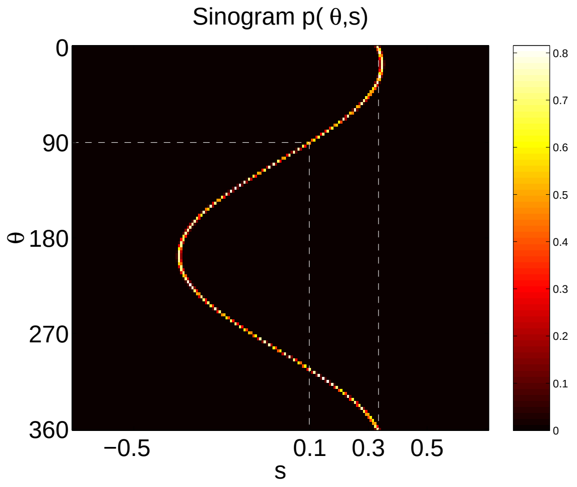
The Sinogram: Properties

Example: Below is an image of $f(x,y) = \delta(x - 0.3346,\ y - 0.0984)$:



The Sinogram: Properties

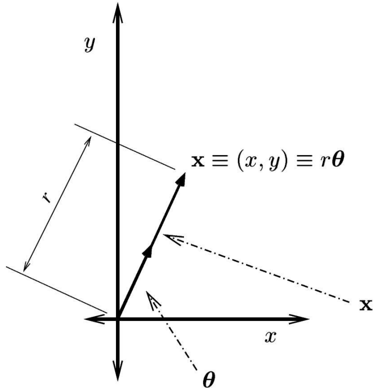
This point is mapped to $s = 0.3346\cos(\theta) + 0.0984\sin(\theta)$:



Radon transform: Properties in N and/or two dimensions

Property	Distribution	Radon transform $\mathbf{R}f$
Linearity	$f(\mathbf{x}) = \sum_i f_i(\mathbf{x})$	$\mathbf{R}f(\mathbf{x}) = \sum_i \mathbf{R}f_i(\mathbf{x})$
Limited support	$f(\mathbf{x}) = 0,\ \lvert x_i\rvert > D/2$	$\mathbf{R}f(\mathbf{x}) = 0,\ \lvert s\rvert > D\sqrt{N}/2$
Symmetry	$f(\mathbf{x})$	$\mathbf{R}f(\mathbf{x}) = p(\boldsymbol{\theta}, s) = p(-\boldsymbol{\theta}, -s)$
	$f(x, y)$	$p(\boldsymbol{\theta}, s) = p_\theta(s) = p_{\theta\pm\pi}(-s)$
Periodicity	$f(x, y)$	$p_\theta(s) = p_{\theta\pm 2k\pi}(s),\ k \in \mathcal{Z}$
Translation	$f(\mathbf{x} - \mathbf{x}_0)$	$p(\boldsymbol{\theta}, s - \mathbf{x}_0 \cdot \boldsymbol{\theta})$
	$f(x - x_0, y - y_0)$	$p(\boldsymbol{\theta}, s - x_0\cos(\theta) - y_0\sin(\theta))$
Rotation	$f(r, \mathbf{T}\boldsymbol{\phi})$	$p(\mathbf{T}\boldsymbol{\theta}, s),\ \mathbf{T}$: rotation matrix
	$f(r, \phi + \theta_0)$	$p_{\theta+\theta_0}(s)$
Scaling	$f(a\mathbf{x})$	$\frac{1}{\lvert a\rvert}p(\boldsymbol{\theta}, as)$
Mass conservation	$M = \int_{\Re^N} f(\mathbf{x})\,d\mathbf{x}$	$M = \int_{-\infty}^{\infty} p(\boldsymbol{\theta}, s)\,ds$

Aside: Review of polar coordinates



It is possible to represent the position of a point in a space of any dimension by specifying a direction unit vector $\boldsymbol{\theta}$ ($\|\boldsymbol{\theta}\| = 1$) and a distance along that vector r. In 2D, we have $x = r\cos(\theta)$ and $y = r\sin(\theta)$ where θ is the angle between $\boldsymbol{\theta}$ and the positive x-axis, measured counterclockwise. The function $\mathbf{x} = r\,\boldsymbol{\theta}_1$ is the equation of a line through the origin having slope $\arctan(\theta_1)$. The variable r tells us how far along $\boldsymbol{\theta}_1$ to go to reach $\mathbf{x}$.

Aside: Review of polar coordinates

When we wish to convert an integral over the x-y plane to polar coordinates, we must introduce the Jacobian determinant into the integral. This maps infinitesimal areas in the x-y plane to the r-θ plane. This determinant is:

$$|\mathbf{J}| = \begin{vmatrix} \frac{\partial x}{\partial r} & \frac{\partial x}{\partial \theta} \\ \frac{\partial y}{\partial r} & \frac{\partial y}{\partial \theta} \end{vmatrix}$$

Since $x = r\ \cos(\theta)$ and $y = r\ \sin(\theta)$ we have:

$$\begin{vmatrix} \cos(\theta) & -r\ \sin(\theta) \\ -\sin(\theta) & r\ \cos(\theta) \end{vmatrix} = r\ \cos^2(\theta) + r\ \sin^2(\theta) = r$$

The tranformation of the integrals is given by:

$$\int_{\Re^2} f(x, y)\,dx\,dy = \int_S \int_0^\infty f(r, \boldsymbol{\theta})\,|\mathbf{J}|\,dr\,d\theta = \int_S \int_0^\infty f(r, \boldsymbol{\theta})\,r\,dr\,d\theta$$

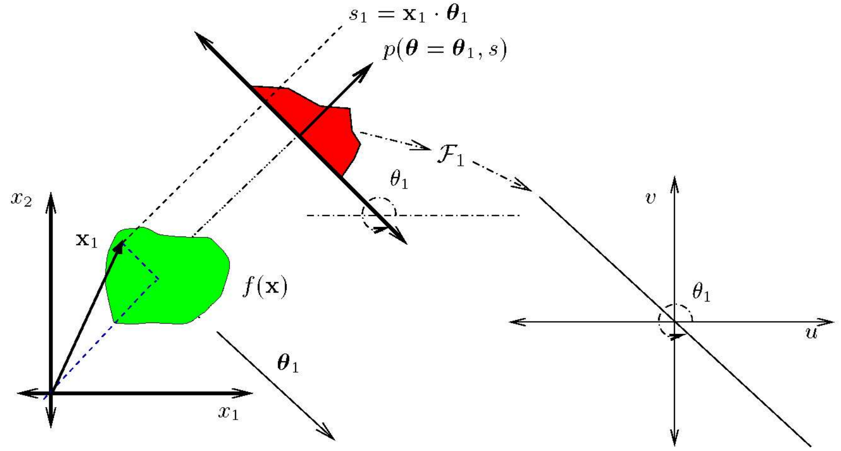
where S is the unit circle.

The Projection Slice Theorem

- The projection slice theorem reveals a very interesting relationship between the Radon and Fourier transforms.
- We will derive this theorem for 1D projections of 2D distributions.

  Theorem: The 1D Fourier transform of the 1D parallel projection $p(\boldsymbol{\theta}, s)$ is equal to the central slice, at an angle θ with respect to the u-axis, of the 2D Fourier transform $F(u, v)$ of the distribution $f(x, y)$.

The Projection Slice Theorem



The Projection Slice Theorem

Proof: The 2D Fourier transform (FT) of $f(x,y) \triangleq f(\mathbf{x})$, $\mathbf{x} \in \Re^2$ is given by:

$$F(\boldsymbol{\xi}) = \int_{\Re^2} f(\mathbf{x})\, \mathrm{e}^{-\jmath 2\pi \mathbf{x}\cdot\boldsymbol{\xi}} d\mathbf{x} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)\, \mathrm{e}^{-\jmath 2\pi(ux+vy)} dx\, dy = F(u,v)$$

where $\boldsymbol{\xi} = [u\ v]^T$. The FT of the projection at angle θ is:

$$\begin{aligned}
P_\theta(w) &= \int_{-\infty}^{\infty} p(\boldsymbol{\theta}, s)\, \mathrm{e}^{-\jmath 2\pi w s}\, ds \\
&= \int_{-\infty}^{\infty} \int_{\Re^2} f(\mathbf{x})\, \delta(\mathbf{x}\cdot\boldsymbol{\theta} - s)\, d\mathbf{x}\ \mathrm{e}^{-\jmath 2\pi w s}\, ds \\
&= \int_{\Re^2} \int_{-\infty}^{\infty} f(\mathbf{x})\, \delta(\mathbf{x}\cdot\boldsymbol{\theta} - s)\ \mathrm{e}^{-\jmath 2\pi w s}\, ds\, d\mathbf{x} \\
&= \int_{\Re^2} f(\mathbf{x})\ \mathrm{e}^{-\jmath 2\pi w\, \mathbf{x}\cdot\boldsymbol{\theta}}\, d\mathbf{x} \\
&= \int_{\Re^2} f(\mathbf{x})\ \mathrm{e}^{-\jmath 2\pi \mathbf{x}\cdot(w\boldsymbol{\theta})}\, d\mathbf{x} = F(\boldsymbol{\xi} = w\,\boldsymbol{\theta})\ \blacksquare.
\end{aligned}$$

The Projection Slice Theorem: Interpretation

We just found that:

$$\mathcal{F}_1\left\{p(\boldsymbol{\theta}, s)\right\} = P_\theta(w) = F(\boldsymbol{\xi} = w\,\boldsymbol{\theta})$$

This means that the 1D Fourier transform of the projection along $\boldsymbol{\theta} = \boldsymbol{\theta}_1$ is equal to the 2D Fourier transform of the distribution $f(\mathbf{x})$ along the vector $\boldsymbol{\theta}_1$. Recall that $\boldsymbol{\xi} = w\,\boldsymbol{\theta}_1$ is the line through the origin along the unit vector $\boldsymbol{\theta}_1$.

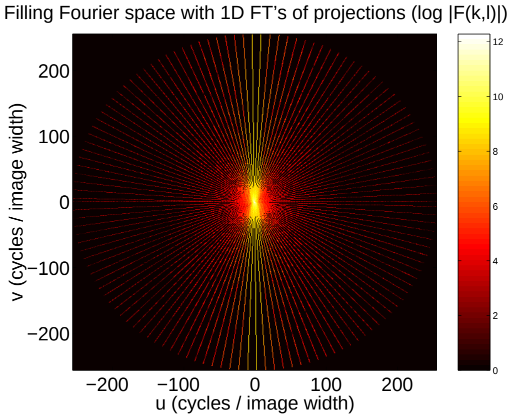
In scalar form we have, equivalently (continuing from the last line of the proof):

$$\begin{aligned}
P_\theta(w) &= \int_{\Re^2} f(\mathbf{x})\ \mathrm{e}^{-\jmath 2\pi \mathbf{x}\cdot(w\boldsymbol{\theta})}\, d\mathbf{x} \\
&= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)\ \mathrm{e}^{-\jmath 2\pi(w\cos(\theta)x + w\sin(\theta)y)}\, dx\, dy \\
&= F(w\cos(\theta), w\sin(\theta)) = F(u,v)\Big|_{u=w\cos(\theta),\ v=w\sin(\theta)}
\end{aligned}$$

The Projection Slice Theorem: Reconstruction

Exercise: Based on the projection slice theorem, find a method to reconstruct $f(x,y)$ given only its projections $p(\boldsymbol{\theta}, s)$.

The Projection Slice Theorem: Reconstruction



In order to use the FFT to transform the discrete version of $F(\boldsymbol{\xi})$, $F(k,l)$, to image space, it is necessary to interpolate the "assigned" values onto a square grid. This is difficult and time-consuming. An alternative is to use a non-uniform DFT, but this is also very slow.

The Projection Slice Theorem: Reconstruction

- In order to obtain $f(x,y)$ from $F(u,v)$, we need an infinite number of projections.
- In practice, only a finite number are available.
- We can use interpolation to find values in between the radial lines on which we have samples of $F(u,v)$, but the interpolation at high frequencies will be less accurate than at low frequencies?

  Why? ______________________________

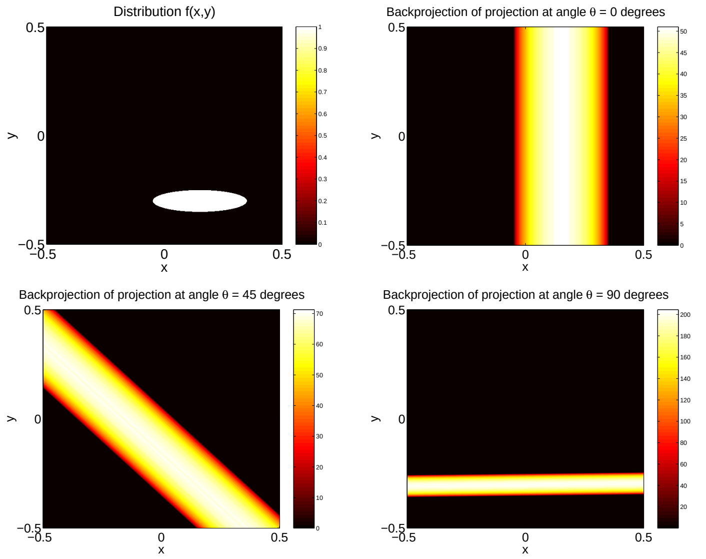
  How will this affect the reconstructed image?

  ____

Practical methods of inverting the Radon transform

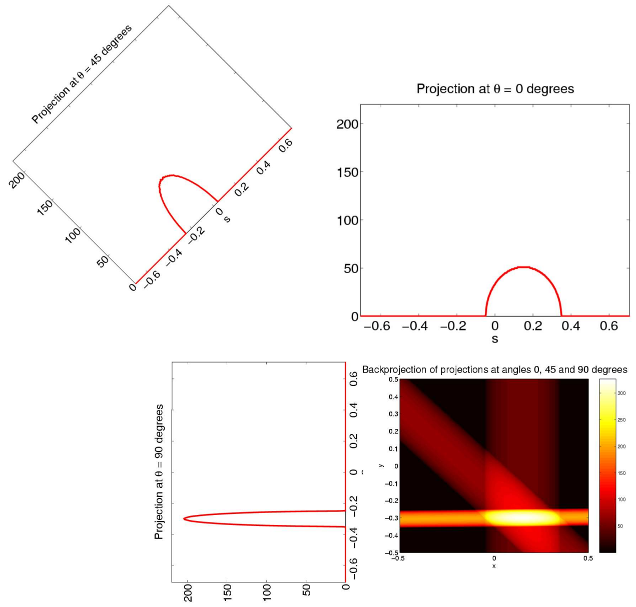
- The projection slice theorem (PST) offers us a method of obtaining a cross-section $f(\mathbf{x})$ from its projections. This is the **inverse** of applying the Radon transform to a distribution $f(\mathbf{x})$ to get its projections $p(\boldsymbol{\theta}, s)$. Consequently, the PST is a method for **inverting the Radon transform**.
- We will see that just by rewriting the projection slice theorem, we can find the most commonly used practical method of performing this inversion.
- Before we do this, we need to understand an operation that projects the projections back into image space. This is called the **backprojection operator.**
- We begin with an example.

The backprojection operator

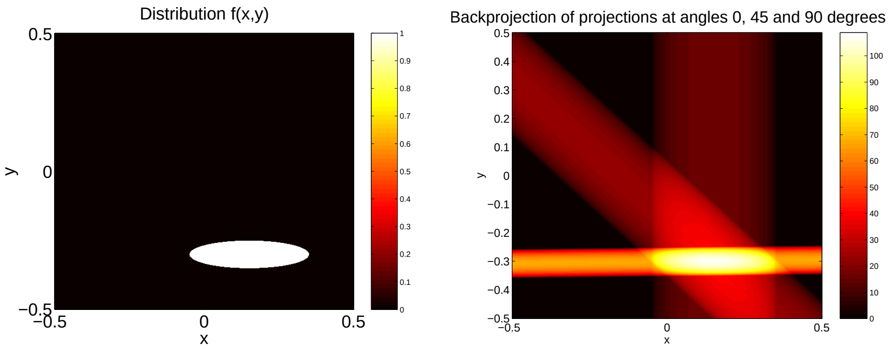


Here we see the individual backprojections of three different projections.

The backprojection operator

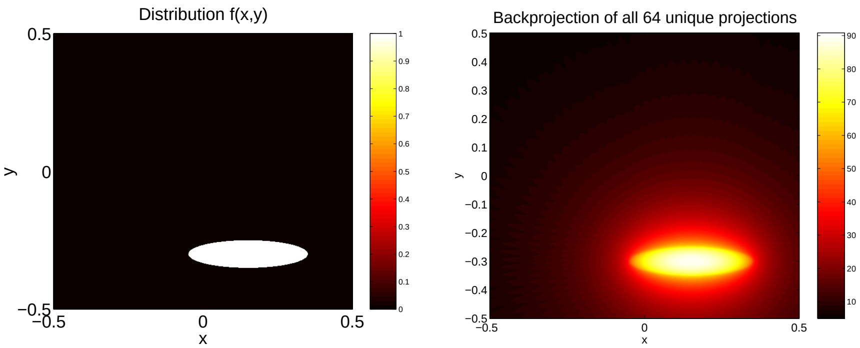


The backprojection operator



After adding the results of only three backprojections, we can get some idea of what the distribution looks like. However, we can see that no matter how many individual backprojections we add, we will still have “streak artifacts”. This is because projections are non-negative, so adding more will not remove the blurring effect of backprojection.

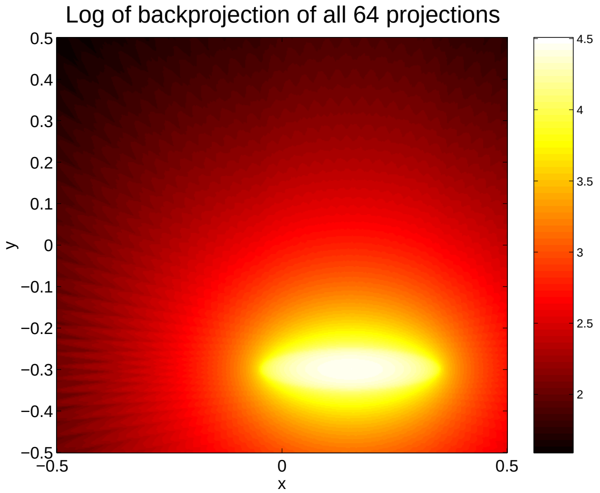
The backprojection operator



After backprojecting all 64 unique projections (for $\theta \in (0, 180)$), we see an elliptical object, but it is blurred and has a much greater intensity than the true $f(\mathbf{x})$.

Where have the streaks gone?

The backprojection operator



Taking the log of the backprojection image, we see the streaks are still there.

The backprojection operator

We now define the backprojection operator mathematically. This operator operates on all unique projections in the interval [0, 180) and smears them out over real space.

$$f_{\mathrm{BP}}(\mathbf{x}) = \mathrm{BP}\{p(\boldsymbol{\theta}, s)\} \triangleq \int_0^{\pi} p(\boldsymbol{\theta}, s)\, d\boldsymbol{\theta}$$

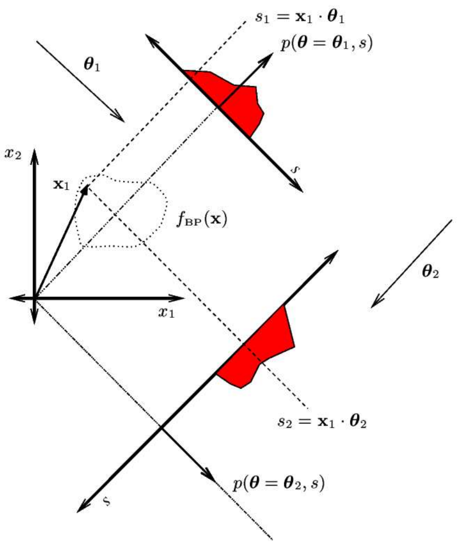
This equation says that the backprojection image $f_{\mathrm{BP}}(\mathbf{x})$ will have a value at point $\mathbf{x}_1$ that can be found by extending the line $s = \mathbf{x}_1 \cdot \boldsymbol{\theta}$ to intersect each projection, recording the value of each projection at the point of intersection, and summing these values.

For a finite number of projections I at angles defined by $\boldsymbol{\theta}_i$ we have:

$$f_{\mathrm{BP}}(\mathbf{x}) = \frac{1}{I} \sum_{i=1}^{I} p(\boldsymbol{\theta}_i, s = \mathbf{x} \cdot \boldsymbol{\theta}_i)$$

We normalize by the number of projections because of the mass conservation property of the Radon transform. In other words, because each projection contains all the "mass" in the distribution $f(\mathbf{x})$.

The backprojection operator


$s_1 = \mathbf{x}_1 \cdot \boldsymbol{\theta}_1$
$p(\boldsymbol{\theta} = \boldsymbol{\theta}_1, s)$
$\boldsymbol{\theta}_1$
s
x_2
$\mathbf{x}_1$
$f_{\mathrm{BP}}(\mathbf{x})$
$\boldsymbol{\theta}_2$
x_1
$s_2 = \mathbf{x}_1 \cdot \boldsymbol{\theta}_2$
$p(\boldsymbol{\theta} = \boldsymbol{\theta}_2, s)$
s

Here two projections contribute to the backprojection image. Thus $f_{\mathrm{BP}}(\mathbf{x}) = \frac{1}{2}\Big[p(\boldsymbol{\theta}_1, s_1 = \mathbf{x} \cdot \boldsymbol{\theta}_1) + p(\boldsymbol{\theta}_2, s_2 = \mathbf{x} \cdot \boldsymbol{\theta}_2)\Big]$.

The backprojection of filtered projections algorithm (BPFP)

- The BPFP algorithm is the most commonly employed tomographic reconstruction algorithm.
- Most texts call it the "filtered backprojection" (FBP) algorithm, but this is a misnomer, as we will see.
- We begin by expressing $f(\mathbf{x})$ in terms of its Fourier transform $F(\boldsymbol{\xi})$. Again, the vector $\boldsymbol{\xi}$ contains the Cartesian vertical and horizontal frequency space coordinates u and v:

$$\boldsymbol{\xi} = \begin{bmatrix} u \\ v \end{bmatrix}.$$

Using the expression for the inverse Fourier transform:

$$f(\mathbf{x}) = \int_{\Re^2} F(\boldsymbol{\xi})\, \mathrm{e}^{\jmath 2\pi \boldsymbol{\xi} \cdot \mathbf{x}}\, d\boldsymbol{\xi}$$

The backprojection of filtered projections algorithm (BPFP)

We now convert to polar coordinates. This is easily achieved. We just define a unit vector $\boldsymbol{\theta}$ that lies along $\boldsymbol{\xi}$ and introduce a scalar w that tells us how far along $\boldsymbol{\theta}$ to go to get to $\boldsymbol{\xi}$. Formally, $\boldsymbol{\xi} = w\,\boldsymbol{\theta}$. We must also introduce the Jacobian for the coordinate change:

$$|\mathbf{J}| = w.$$

into the integral.
After the coordinate change we have:

$$\begin{aligned} f(\mathbf{x}) &= \int_S \int_0^{\infty} F(w\,\boldsymbol{\theta})\, \mathrm{e}^{\jmath 2\pi w \boldsymbol{\theta} \cdot \mathbf{x}}\, w\, dw\, d\boldsymbol{\theta} \\ &= \int_S \int_0^{\infty} P_{\boldsymbol{\theta}}(w)\, \mathrm{e}^{\jmath 2\pi w \boldsymbol{\theta} \cdot \mathbf{x}}\, w\, dw\, d\boldsymbol{\theta} \end{aligned}$$

where we have used the projection slice theorem to make the substitution:

$$F(w\,\boldsymbol{\theta}) = P_{\boldsymbol{\theta}}(w).$$

The backprojection of filtered projections algorithm (BPFP)

Now, the outer integral sums over all direction vectors $\boldsymbol{\theta}$ around the unit circle S. However, we know that the Radon transform has the symmetry property:

$$p(\boldsymbol{\theta}, s) = p(-\boldsymbol{\theta}, -s)$$

Consequently, we must also have:

$$P_{\boldsymbol{\theta}}(w) = P_{-\boldsymbol{\theta}}(w).$$

Invoking the symmetry property of the FT: $X(-\omega) = X^*(\omega)$ we have, in addition:

$$P_{\boldsymbol{\theta}}(w) = P_{-\boldsymbol{\theta}}(-w).$$

The backprojection of filtered projections algorithm (BPFP)

Therefore, we can extend the limits on the inner integral to the entire real axis and divide the result by two:

$$\begin{aligned} f(\mathbf{x}) &= \int_S \int_0^\infty P_{\boldsymbol{\theta}}(w)\, \mathrm{e}^{\jmath 2\pi w \boldsymbol{\theta}\cdot\mathbf{x}}\, w\, dw\, d\boldsymbol{\theta} \\ &= \frac{1}{2} \int_S \int_{-\infty}^\infty P_{\boldsymbol{\theta}}(w)\, \mathrm{e}^{\jmath 2\pi w \boldsymbol{\theta}\cdot\mathbf{x}}\, |w|\, dw\, d\boldsymbol{\theta} \\ &= \frac{1}{2} \int_S \int_{-\infty}^\infty P_{\boldsymbol{\theta}}(w)\, |w|\, \mathrm{e}^{\jmath 2\pi w \boldsymbol{\theta}\cdot\mathbf{x}}\, dw\, d\boldsymbol{\theta} \end{aligned}$$

Similarly, we can integrate over only the first two quadrants of the unit circle, and multiply the integral by two to compensate:

$$= \quad 2 \times \frac{1}{2} \int_0^\pi \int_{-\infty}^\infty P_{\boldsymbol{\theta}}(w)\, |w|\, \mathrm{e}^{\jmath 2\pi w s}\, dw\, d\theta$$

(we have also now made the substitution $s = \boldsymbol{\theta} \cdot \mathbf{x}$).

The backprojection of filtered projections algorithm (BPFP)

Let's take a look at the final equation:

$$f(\mathbf{x}) \quad = \quad \int_0^\pi \int_{-\infty}^\infty P_{\boldsymbol{\theta}}(w)\, |w|\, \mathrm{e}^{\jmath 2\pi w s}\, dw\, d\theta$$

Let's define $H(w) = |w|$, and $G(w) = P_{\boldsymbol{\theta}}(w)$.

$$f(\mathbf{x}) \quad = \quad \int_0^\pi \int_{-\infty}^\infty G(w)\, H(w)\, \mathrm{e}^{\jmath 2\pi w s}\, dw\, d\theta$$

Taking the Fourier transform of both sides gives:

$$F(u, v) \quad = \quad \int_0^\pi G(w)\, H(w)\, d\theta$$

Now, we recognise the integral over θ as the backprojection operator, so:

$$F(u, v) \quad = \quad \mathrm{BP}\{G(w)\, H(w)\}$$

The BPFP algorithm

The equation

$$F(u, v) \quad = \quad \mathrm{BP}\{G(w)\, H(w)\}$$

represents the Fourier domain filtering of the projections $p(\boldsymbol{\theta}, s)$ by the filter $G(w) = |w|$. If $G(w)$ was equal to 1 at all frequencies w, we would get:

$$\begin{aligned} f'(x, y) &= \mathrm{BP}\Big\{\mathcal{F}_{-1}\Big\{G(w)\Big\}\Big\} \\ &= \mathrm{BP}\{p(\boldsymbol{\theta}, s)\} \\ &= f_{\mathrm{BP}}(x, y) \end{aligned}$$

which is the blurred backprojection image (we can interchange the Fourier and backprojection operators because they are linear). The derivation we have just completed tells us that if we filter the FTs of the projections with the filter $H(w) = |w|$, **before backprojecting** we will get back the true distribution $f(x, y)$.

The BPFP algorithm: The ramp filter

- The filter that we must apply to the projections $H(w) = |w|$ is a ramp in frequency.
- We know by the convolution theorem that since:

$$F(u,v) \quad = \quad \mathrm{BP}\{G(w)\,H(w)\}$$

it must be that:

$$f(x,y) \quad = \quad \mathrm{BP}\{g(s) * h(s)\} = \mathrm{BP}\{p(\boldsymbol{\theta}, s) * h(s)\}$$

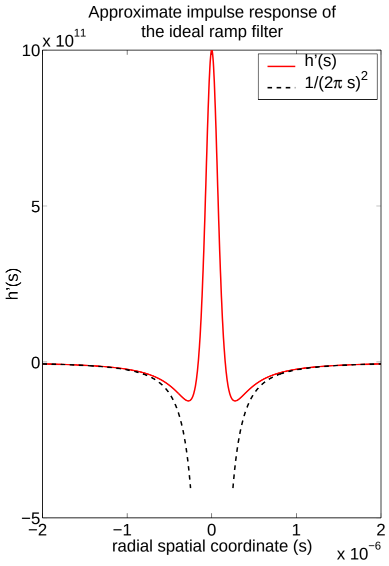
- What does $h(s)$ look like? Because $|w|$ is not square-integrable, it does not have a Fourier transform. Instead, we can find the FT of a square-integrable function that becomes $|w|$ asymptotically:

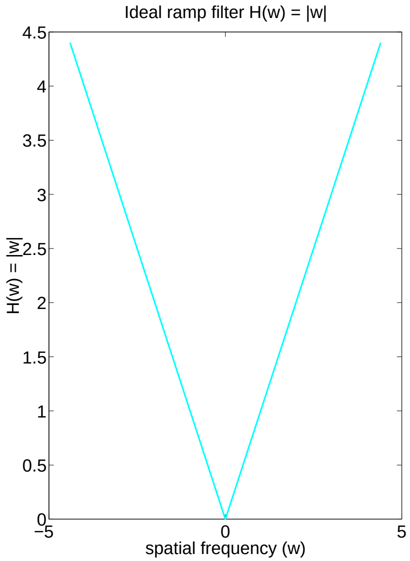
$$H(w) = \lim_{\epsilon \to 0} |w|\, \mathrm{e}^{-\epsilon\,|w|}$$

The FT of this function is:

$$h'(s) = \frac{\epsilon^2 - (2\pi s)^2}{(\epsilon^2 - (2\pi s)^2)^2} \approx -\frac{1}{(2\pi s)^2} \text{ for large } s$$

The BPFP algorithm: The ramp filter





The BPFP algorithm: The ramp filter

We can improve upon this approximate expression for the impulse response.

- First, we reexpress the ramp filter as:

$$H(w) = |w| = w\,\mathrm{sgn}(w)\,\frac{2\pi \jmath}{2\pi \jmath}$$

where

$$\mathrm{sgn}(w) = \begin{cases} 1 & \text{for } w > 0 \\ -1 & \text{for } w < 0 \end{cases}$$

- Then we substitute this expression into the BPFP equation:

$$f(\mathbf{x}) = \int_0^{\pi} \int_{-\infty}^{\infty} 2\pi \jmath\, w P_{\boldsymbol{\theta}}(w)\, \frac{\mathrm{sgn}(w)}{2\pi \jmath}\, \mathrm{e}^{\,\jmath 2\pi w s}\, dw\, d\theta$$

The BPFP algorithm: The ramp filter

- Applying the convolution theorem, and taking inverse Fourier transforms we get:

$$\begin{aligned} f(\mathbf{x}) &= \int_0^{\pi} \mathcal{F}_1^{-1}\Big\{2\pi \jmath\, w P_{\boldsymbol{\theta}}(w)\Big\} * \mathcal{F}_1^{-1}\Big\{\frac{\mathrm{sgn}(w)}{2\pi \jmath}\Big\} d\theta \\ &= \int_0^{\pi} \frac{\partial p(\boldsymbol{\theta}, s)}{\partial s} * \frac{1}{2\pi^2 s}\, d\theta \end{aligned}$$

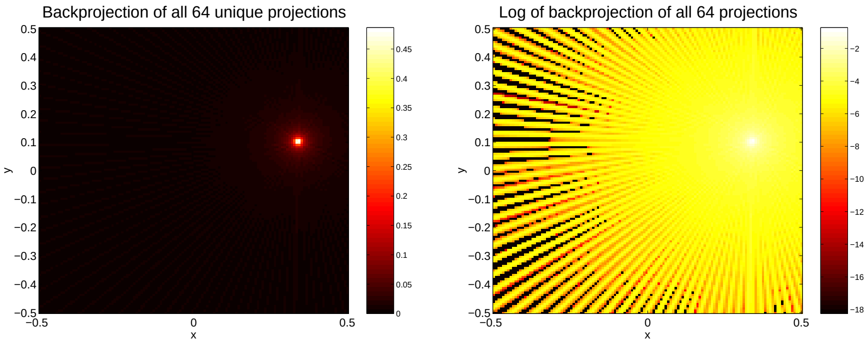
- For any function $f(s)$:

$$g(s) = \frac{1}{2\pi^2 s} * f(s)$$

is the **Hilbert transform** of $f(s)$.

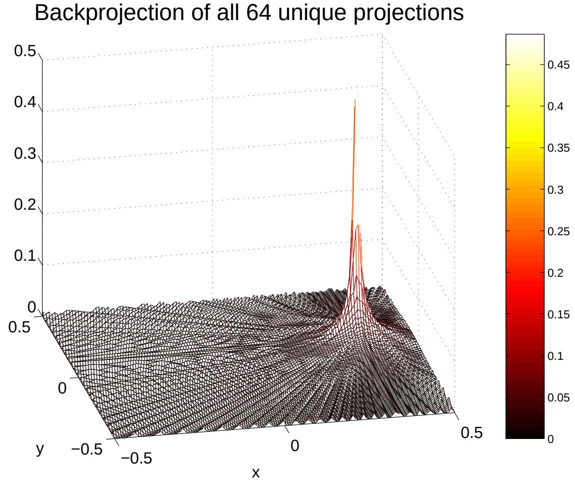
- Filtering a projection $p_{\boldsymbol{\theta}}(s)$ with the ramp filter in the frequency domain is thus equivalent to taking the Hilbert transform of the derivative of same projection with respect to s .

The backprojection operator



We return briefly to the example of a distribution consisting of a single point. When we backproject the projections of this point, we get the impulse response of the backprojection operator. To get back the original impulse $f(x, y) = \delta(x - 0.3346,\ y - 0.0984)$, we need to filter the projections by the inverse of this impulse response.

The backprojection operator



The impulse response of the backprojection operator is non-negative at all points. Thus, it is a low-pass (averaging) filter. To undo the effects of this filter, we need a high-pass filter. The latter must necessarily have some negative values.

The BPFP algorithm: Practical implementation

- The ramp filter has infinite support in the frequency domain. It is therefore not possible to implement this filter in reality.
- In practical imaging, the imaging equipment will sample the projections at a sample spacing of X distance units. This corresponds to a sampling frequency of $W = 1/\Delta s$ samples/distance unit.
- The projections are thus **bandlimited** to $W/2$ cycles/distance unit.
- Because of the bandlimited nature of the projections, we can truncate the ramp filter at $W/2$ cycles/distance unit.
- If the distribution contains fine detail that is smaller than the sampling interval, how can we prevent this detail from being aliased down to lower frequencies?

  ____.

- If aliasing occurs, will the projections still be bandlimited?

  ____

The BPFP algorithm: Practical implementation

- The truncated ramp filter, also called the **Ram-Lak filter**, is defined as:

$$H(w) = \begin{cases} |w| & \text{for } |w| < W \\ 0 & \text{for otherwise} \end{cases}$$

and can alternatively be expressed as:

$$H(w) = \frac{W}{2}\mathrm{rect}\left(\frac{w}{W}\right) - \frac{W}{2}\mathrm{tri}\left(\frac{2w}{W}\right)$$

where:

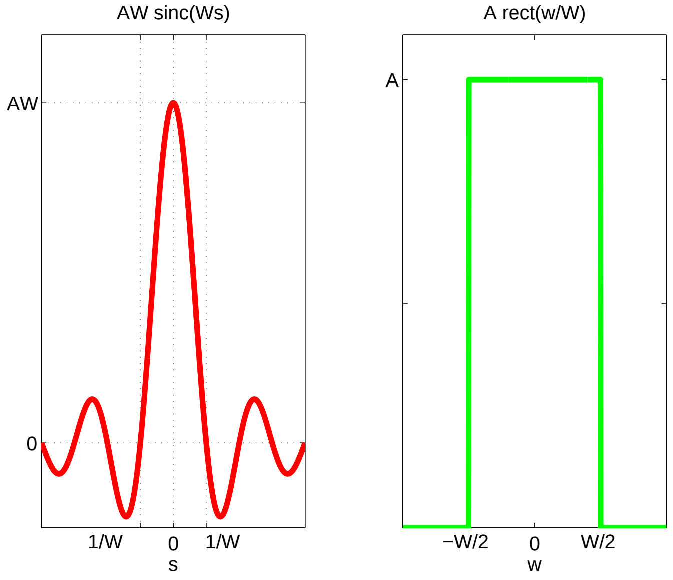
$$\mathrm{tri}(w) = \begin{cases} 1 - \frac{|w|}{W} & \text{for } |w| < W \\ 0 & \text{for otherwise} \end{cases}$$

- The relevant FT pairs are:

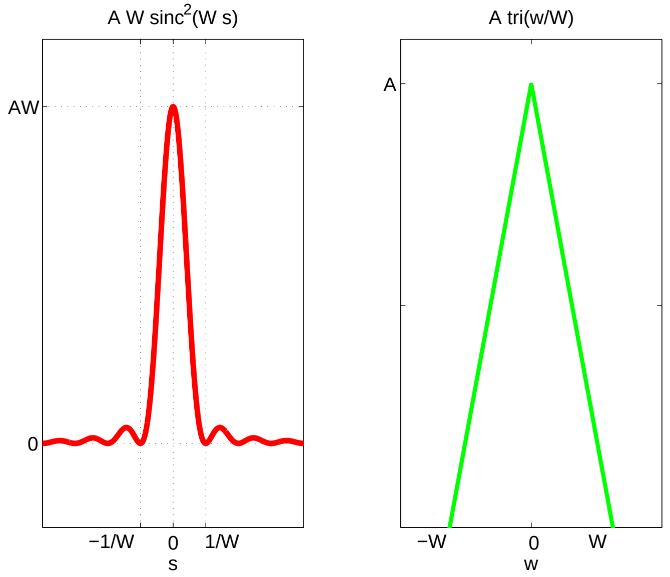
$$A\,\mathrm{rect}\left(\frac{w}{W}\right) \rightleftarrows AW\mathrm{sinc}(Ws)$$

$$A\,\mathrm{tri}\left(\frac{w}{W}\right) \rightleftarrows AW\mathrm{sinc}^2(Ws)$$

Aside: Definition and FT of the rect function



Aside: Definition and FT of the triangle function



The BPFP algorithm: Practical implementation

- Introducing the linear and axis scaling factors we have:

$$\frac{W}{2}\,\mathrm{rect}\left(\frac{w}{W}\right) \rightleftarrows \frac{W^2}{2}\mathrm{sinc}(Ws)$$

$$\frac{W}{2}\,\mathrm{tri}\left(\frac{2w}{W}\right) \rightleftarrows \frac{W^2}{4}\mathrm{sinc}^2\left(\frac{W}{2}s\right)$$

- Taking the inverse FT of $H(w)$ gives the corresponding impulse response as:

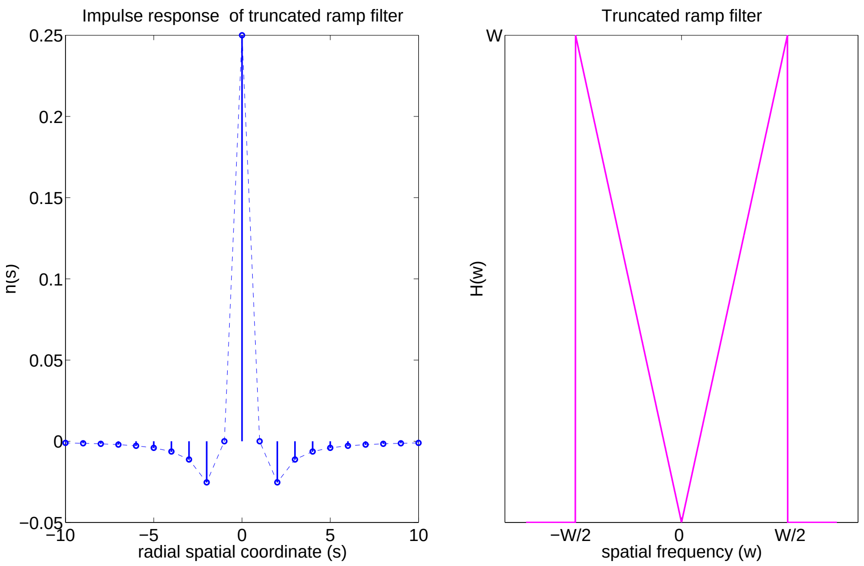
$$h(s) = \frac{1}{2}W^2\mathrm{sinc}(Ws) - \frac{1}{4}W^2\mathrm{sinc}^2\left(\frac{W}{2}s\right);$$

The BPFP algorithm: Practical implementation

- To get the discrete form of $h(s)$, the impulse response of the truncated ramp filter, we set the sampling interval to $\Delta s = 1$, and evaluate $h(s)$ at $s = n\Delta s$. This gives:

$$h[n] = \begin{cases} \frac{1}{4} & n = 0 \\ 0 & n \text{ even} \\ -\frac{1}{n^2\,\pi^2} & n \text{ odd} \end{cases}$$

The BPFP algorithm: Practical implementation



The BPFP algorithm: Practical implementation

- By inspection of the shape of the truncated ramp filter, anticipate two of its shortcomings:

____

____

- What method that we studied earlier in this course would be useful for improving the truncated ramp filter?

____

- What would we sacrifice if we effected this modification?

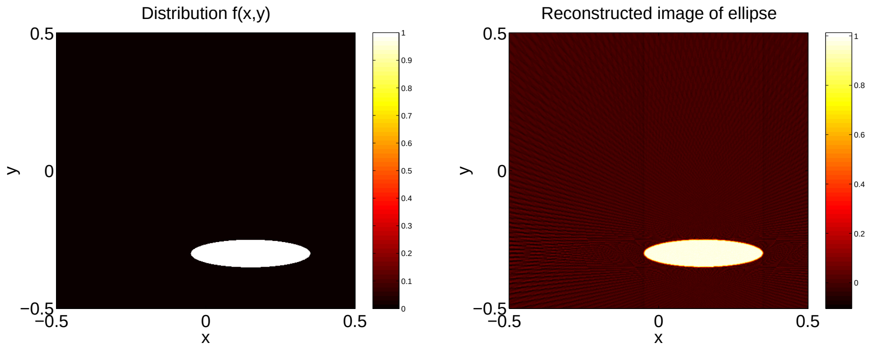
____

____

The BPFP algorithm: Practical implementation

Show schematically the flow of information in the BPFP algorithm. Include schematics for both real space and frequency domain implementations.

The BPFP algorithm: Example



We see that the reconstructed image, $\tilde{f}[m,n]$ is a far better approximation to the original than was the backprojection image. Note that the intensity of the ellipse is very close to the correct amplitude. Owing to discretization, the finite number of projections and filter modification, BPFP images always have some streak artifacts. In this example, the Ram-Lak filter was multiplied by a von Hahn window to reduce noise amplification and ringing.

The BPFP algorithm: Practical implementation

Summary of ramp filter variants:

Ram-Lak Truncated ramp

Shepp-Logan Ram-Lak times sinc function

Lowpass cosine Ram-Lak times cosine

Hamming Ram-Lak times Hamming window

von Hahn Ram-Lak times von Hahn window

See p. 449 of Jain (p. 24 in the reader) for details.

The BPFP algorithm: “Black box” implementation in Matlab

```
% This program draws a solid ellipse in
% image space, then projects this ellipse
% into projection space using Matlab’s radon().
% The image is then reconstructed using Matlab’s
% iradon()

% set ellipse parameters

a2 = .2^2; % square of 1st semiaxis radius
b2 = .05^2; % square of 2nd semiaxis radius
ori = 0; % orientation in degrees
x = 0.15 % position of x center
y = -0.3; % position of y center
intens = 1; % intensity
```

```
N = 512; % image side dimension
imDim = 0.5; % true image halfwidth

% make Cartesian grid to support ellipse
xAx = linspace(-imDim, imDim, N);
yAx = fliplr(xAx);
[xMesh, yMesh] = meshgrid(xAx, yAx);

% call custom function that fills an image with solid ellipses
[im, ind] = solidellipses(a2,b2,ori,x,y, intens, xMesh, yMesh);

numProj = 64; % set number of projections to take

% make projection angle axis for unique projections
theta = linspace(0,180-180/numProj, numProj);

% radon() is a Matlab built-in that performs 2D Radon transform
% is returns the transpose of the sinogram
```

```
[pT, sAxPre] = radon(im, theta);

% make radial Radon axis and scale to image dimension
sAx = sAxPre / max(sAxPre) * sqrt(2) * imDim;

% reconstruct image using Hahn filter and
% nearest neighbor interpolation to cope
% with the discretization
imRec = iradon(pT, theta, ’nearest’, ’Hann’);
```

Making images of solid ellipses in Matlab

```
% function [im, pixRegInd] = solidellipses(a2,b2,ori,x,y,
% intensity, xMesh, yMesh)
%
% This function places solid ellipses inside an image.
% It returns an image im and the indices of all points
% inside each ellipse

function [im, pixRegInd] = solidellipses(a2,b2,ori,x,y, ...
                                     intensity, xMesh,yMesh)

% If only one intensity value is given, make all ellipses
% have this intensity
if length(intensity) < length(a2)
  intensity = repmat(intensity, length(a2));
end
```

```
% initialize image
im = zeros(size(xMesh));

for i = 1:length(a2)
  imPre = solidellipse(a2(i),b2(i),ori(i),x(i),y(i),...
                       intensity(i),xMesh,yMesh);
  pixRegInd{i} = find(imPre==intensity(i));
  % add this ellipse image to the output image
  im = im + imPre;
end;
```

Making images of solid ellipses in Matlab

```
% function [im,ellInd] = solidellipse(a2,b2,ori,x,y,intensity,
% xMesh,yMesh)
%
% This function places a single solid ellipse inside an image

function [im,ellInd] = solidellipse(a2,b2,ori,x,y,...
                                    intensity, xMesh, yMesh)

% initialize image

im = zeros(size(xMesh));

% translate origin to center of ellipse

xMt = xMesh - x;
yMt = yMesh - y;
```

```
% find points inside ellipse

ellInd = find( (xMt*cos(ori) - yMt*sin(ori)).^2 / a2 + ...
               (xMt*sin(ori) + yMt*cos(ori)).^2 / b2 <= 1);

% set all points inside ellipse equal to intensity value
im(ellInd) = ones(size(ellInd)) * intensity;
```

Implementing the discrete backprojection operator in Matlab

```
% This program takes the projections of an ellipse
% and backprojects them. It uses a custom
% Matlab function backpro().
% It assumes the original image im and the transposed
% sinogram pT exist in the Matlab workspace

% initialize empty backprojection image
backIm = zeros(size(im));

% make an index vector for each projection
ind = 1:round(length(theta)/2);

% assign sinogram
p = pT.';
```

```
imBP=[]; % make empty matrix to store
         % backprojections of single projections

for i = 1:length(ind)
  thisInd = ind(i);
  thisTheta = theta(thisInd);
  % find backprojection from projection i at angle i
  imBP(:,:,i) = backpro(backIm, p(thisInd,:), thisTheta);
end

% sum all the backprojections and normalize by number
% of projections (each projection contains the entire
% mass of the distribution)
imBPSum = sum(imBP,3)/length(ind);
```

Implementing the discrete backprojection operator in Matlab

Slow, easier to understand version of operator implementation:

```
function back2 = backpro(back, proj, theta)
% back2 = backpro(back, proj, theta)
%
% This function back projects a single projection vector 'proj'
%  obtained at an angle 'theta' into a real space backprojection array
%  'back' with interpolation between samples in the projection; it returns
%  the modified backproject array.  The center of rotation is assumed to
%  be at the center of the backprojection array and the center of the
%  projection vector.  The back projection array should be initialized
%  to zero before backproj is called with the first projection vector.
%  The angle is measured counterclockwise and zero corresponds to a vertical
%  (column) sum of the backprojection matrix; the image is assumed to
%  be viewed using the matlab image function which displays the lowest-
%  numbered row at the top of the image.
%
%     NOTE:  The function backproj produces the same results and is
%  much faster; it should be used instead of this function.

%  Summary of variables used
%   x, y Coordinates in backprojection array
```

```
%   u Coordinate in projection array
%   iu Integer part of u
%   fu Fractional part of u
%   xc, yc Center of backprojection array
%   uc Center of projection array
%   sint, cost Sine and cosine of theta

%  Find sizes and compute some useful parameters
[nx,ny] = size(back);
m = max(size(proj));
xc = (nx+1)/2.0;
yc = (ny+1)/2.0;
uc = (m+1)/2.0;
sint = sin (theta*pi/180.0);
cost = cos (theta*pi/180.0);

% Loop over pixels in backprojection array
back2 = back;
for x = 1:nx
   for y = 1:ny
      % Find in projection, interpolate, and add to backprojection
      u = uc + (xc-x)*sint + (y-yc)*cost;
      iu = floor(u);
      fu = u - iu;
      if 1 <= iu & iu < m
```

```
         back2(x,y) = back2(x,y) + (1.0-fu)*proj(iu) + fu*proj(iu+1);
      elseif iu == m
         back2(x,y) = back2(x,y) + (1.0-fu)*proj(iu);
      end
   end
end
```

Implementing the discrete backprojection operator in Matlab

Faster, more difficult to understand version of operator implementation:

```
function back2 = backproj (back, proj, theta)
% back2 = backproj(back, proj, theta)
%
% This function back projects a single projection vector 'proj'
%  obtained at an angle 'theta' into a real space backprojection array
%  'back' with interpolation between samples in the projection; it returns
%  the modified backproject array.  The center of rotation is assumed to
%  be at the center of the backprojection array and the center of the
%  projection vector.  The back projection array should be initialized
%  to zero before backproj is called with the first projection vector.
%  The angle is measured counterclockwise and zero corresponds to a vertical
%  (column) sum of the backprojection matrix; the image is assumed to
%  be viewed using the matlab image function which displays the lowest-
%  numbered row at the top of the image.
%
%     Here is a sample program using the backproj function, where nback
%  is the desired size of the backprojection array and dangle is the
%  angle increment between projections.
%
```

```
%  [nang, nbin] = size(sinogram);
%  back = zeros(nback, nback);
%  for ang = 1:nang
%     back = backproj(back, sinogram(ang,:), (ang-1)*dangle);
%     end

%  This version of backprojection has been somewhat optimized for speed
%  and is thus somewhat difficult to understand;  the alternate version
%  backpro.m is slower but easier to understand.

%  Summary of variables used
%   x, y Coordinates in backprojection array
%   u Coordinate in projection array
%   iu Integer part of u
%   fu Fractional part of u
%   xc, yc Center of backprojection array
%   uc Center of projection array
%   sint, cost Sine and cosine of theta

%  Find sizes and compute some useful parameters
[nx,ny] = size(back);
m = max(size(proj));
proj = [0; proj(:); 0; 0];
xc = (nx+1)/2.0;
```

```
yc = (ny+1)/2.0;
uc = (m+1)/2.0;
sint = sin (theta*pi/180.0);
cost = cos (theta*pi/180.0);

% Do the backprojection by columns of the backprojection array
back2 = back;
x = 1:nx;
for y = 1:ny;
   u = uc + (xc-x)*sint + (y-yc)*cost;
   iu = floor(u);
   fu = u - iu;
   iu = max(1, iu+1);
   iu = min(iu, m+2);
   back2(:,y) = back2(:,y) + (1.0-fu)'.*proj(iu) + fu'.*proj(iu+1);
   end
end
```